Tilburg University
The Netherlands

SilverPlatter Information Ltd
United Kingdom

European University Institute
Italy

Universitat Autònoma de Barcelona
Spain

London School of Economics and Political Science
United Kingdom

DECOMATE-II
LIB-5672/B

Developing the European Digital Library for Economics

Analysis and Design Advanced Access Functionality

Editor/Author(s)	Jeroen Hoppenbrouwers
Document Number	TU-1998-3
Date	December 14, 1998
Scheduled	November 30, 1998
Task	4.1.4
Deliverable	4.1
Status	Final
Version	1.0
Keywords	Information Space Browser, Lexicon, De-duplication, Document Tracer
Abstract	This document describes the various modules that provide the Advanced Access facilities to the Decomate-II system. It contains the argumentation for the design and the implementation intentions as far as currently can be decided.

Contents

Chapter 1

Executive Summary

The Advanced Information Space Browser (ISB) provides for 'conceptual' access to the bibliographical databases, as opposed to the traditional keyword-based 'search engine' approach supported by the rest of the Decomate II system. Conceptual browsing promises to help users in transparently building database queries that are more effective in both document retrieval precision and recall.

The design includes a user front-end, containing the graphical interface, which runs in a standard Web browser using a Java applet, plus a back-end which contains a large conceptual database and some Decomate-specific optimizations. Together these modules deliver a single access point for both conceptual browsing and query result presentation, including relevance feedback and relevance ranking. The conceptual database is structured in such a way that it can be distributed over the various cooperating libraries in the project, to enable each library to build (parts) of the conceptual network independent of the others. When required, the resulting partial networks can still be reconnected to form a single, federated conceptual space, without the need for careful mutual tuning.

Due to the inherent innovative character of Workpackage 4, it is difficult to design the ISB in the same detail as the rest of the Decomate II system at this moment in the project. We have a fair overview of what we want to have accomplished at the end of the project, but cannot exactly point out the implications of all design decisions yet. The research involved with this workpackage will have consequences for the design and the implementation of the ISB that only will surface further down the project. However, care has been taken to isolate these possible consequences from the rest of the Decomate II system—the ISB is for the largest part an independent module *on top* of the Decomate II system, and backtracking some phases of the design will not influence work in progress in other workpackages.

Chapter 2

State of the Art in Information Retrieval

Much work in the area of indexing and retrieval concentrates on constructing effective and efficient algorithms to find a set of 'interesting' documents in a large collection, given a user query of some sort. So-called *full text query engines* are used to mechanically select documents out of a collection, either by Boolean keyword matching or according to statistical computations (mainly clustering techniques). Boolean engines usually return all and only documents that exactly match the query. The statistic retrieval engines use various combinations of text metrics in order to predict the document's value in terms of the user query (Salton and McGill, 1983; Salton, 1991). This is called *relevance ranking*. Usually only a limited amount of documents (e.g., the top 100 best matches) is returned to the user.

2.1 The Keyword Barrier

Because most relevance ranking algorithms are based on *textual* data, i.e., actual word forms (strings) without any sense interpretation, they tend to perform generally mediocre (Woods, 1997; Shaw et al., 1997; Blair and Maron, 1985). The same problem appears in Boolean engines. Blair and Maron list many problems with keyword-based searches, and most of their findings fall into two categories:

- Phraseology: synonyms, slang, and jargon terms obscure the meaning of the text, making it very difficult to locate by keyword approaches.
- Granularity: as database size increases, increasingly fine-grained searches are necessary. Retrieval techniques that only consider the presence or absence of words cannot distinguish different relationships between the same

> words, and retrieve far more documents than the user considers relevant (Carbonell and Thomason, 1986).

The crucial problem of current information retrieval technology is that systems relying solely on the presence or absence of a word are inherently limited in their ability to distinguish relevant and irrelevant texts. Word-based systems that cannot deal with synonymy, polysemy, metaphor, and the other complications of natural (uncontrolled) language must have some upper bound on retrieval performance, the *keyword barrier* (Maulding, 1991).

One way of breaking the keyword barrier would be to add semantic knowledge about the document content to the database, so that the above (mostly syntactic and lexical) problems are avoided. Unfortunately, this is not feasible in the context of the Decomate II project, since the databases and query engines are given. Our only chance to break the barrier is to enrich the query that is sent to the database back ends, using a 'knowledgeable' intermediary that *does* possess semantic knowledge, and treat the returned results with a comparable device for additional filtering, to provide relevance feedback, and to add relevance ranking.

The usage of so-called *search intermediaries*, either human (librarians or documentalists) or mechanical, has been proposed both to shield the user from the technical aspects of the query engine and to provide extra background information about the document's domain (Wiesman, 1998). The objective of these systems is to be helpfully responsive to a spontaneous description of the information required, minimizing the need for an information seeker to engage in repeated query reformulation in order to discover the exact terminology that will retrieve the information required (Hoppenbrouwers, 1998; Woods, 1997, p.12).

Therefore our work focuses on various difficulties in applying such a search intermediary to an *existing* library system, based on Boolean keyword retrieval from an uncontrolled vocabulary.

2.2 Conceptual Queries and Knowledge Navigation

Parsaye *et al.* (1989, Ch.6) distinguish five kinds of knowledge that users need to have in order to successfully compose conceptual queries for information retrieval systems: procedural system knowledge, strategic search request formulation knowledge, indexing policy knowledge, search strategy knowledge, and domain knowledge. Of these five, the first four types of knowledge can be leveraged by suitable user training and system manuals. Domain knowledge, however, is typically not suitable to be presented in a manual or through simple training, and when extensive education is not an option, some form of extra help must be supplied by the system. Additionally, extra domain knowledge is required to formu-

late good queries, especially with complex information needs (Hoppenbrouwers, 1998; Bodner and Song, 1996; Howard, 1992).

Conceptual queries involve *knowledge navigation*. There are two basic types of knowledge navigation: *searching* and *browsing* (Wiesman, 1998, p.7). Since searching implies that the user already knows what to look for, this is no solution to the lack of domain knowledge. In such a situation, complete *topic-level concept browsing* is required (Hoppenbrouwers, 1998; Papazoglou, 1997). Interactive term suggestion, where the system suggests terms for the user to choose, can also significantly enhance retrieval effectiveness (Schatz et al., 1996; Papazoglou et al., 1998).

Basic idea behind such a system is to provide the user with an 'information space' (IS) through which (s)he can freely browse, and which gives a fair indication of the *distribution of concept terminology* over the domain. Although the IS should have a reasonable overlap with the actual database, it only serves as a guide: actual queries still are served by the underlying original databases. This means that the concept space browser is allowed to be slightly out of date without significantly affecting the final query result.

In an extreme case, the conceptual space could be very extensive with a very limited document database, a situation comparable with an experienced librarian or documentalist who starts up a new, therefore small, library. The other extreme, a small conceptual space with a large collection of documents, represents a well-stocked library with few support people who know what the documents actually are about.

2.2.1 Terminology Navigation

Placing the user somewhere in the conceptual network to start browsing is not trivial. A common way of finding a suitable starting place in keyword systems is by having the user enter an initial query. However, as argued above, there usually are more places in the same network that warrant attention. Moreover, it is not a good idea to restrict database queries to only the controlled terminology in the network. After all, most entries in the database consist of free natural language, especially if the database contains abstracts and/or full text.[1] When queries are only allowed on keywords from the restricted network, they will in many cases miss important documents. On top, the initial user suggestion itself might contain important terms that are not obvious in the domain. This implies that a good conceptual network should contain as many terms as possible, and be not restricted to 'preferred' terminology (although some concepts certainly may be flagged as

[1] We assume that the database is indexed using a full-text engine, not by manually assigning keywords from a restricted vocabulary.

'preferred' for explicit keyword assignments to database entries). Even words not traditionally associated with a given semantic field should be included to enable intuitive associations to be made.

For example, a user looking for generic 'changes in the deficit' would expect a document to turn up which contains 'Last year's reductions in tax rates are part of the reason for the deficits, as are the administration's plans for a sustained military buildup.' While the query and the passage convey similar ideas, the wording in each is different, a typical case of the paraphrase problem. 'Change' is a more generic variant of both 'reduction' and 'increase,' but neither of the three would be expected in any pure economics thesaurus. Without these terms, either far too many documents would show up, or none at all (Woods, 1997).

This means that the network presentation in the user interface must be dynamic, and must allow navigation in several places at the same time, 'connecting' parts of the network that are not connected by default (Cooper and Byrd, 1997). Additionally, there are technologies which can map a user-provided term into the thesaurus even if the thesaurus does not contain that term explicitly (Woods, 1997). This can help to position the user on the semantic map in several promising starting places, plus to modify the map itself when there is need to do so.

Chapter 3

The Decomate II Architecture

The Decomate II Library System aims at a Web-based, single-point, uniform user interface to a multitude of (possibly distributed) databases. A single user query, usually a set of keywords, is transparently mapped to all connected databases, each with its own query language, data schema, and content. The individual query results are merged together by the system, ordered, and presented to the user in a suitable format.

See Figure 3.1 for an illustration of the Decomate II abstract architecture. The standard query system is directly from a user's browser to the Broker which acts as a Web server. The Broker channels and coordinates all user requests. This is necessary since the Web protocol (HTTP) is stateless and does not allow for session memory. It would be very difficult to implement good user identification and a proper navigation interface without session memory.

Part of the Broker is the *Multi Protocol Server*, or MPS, the module that speaks several protocols to a diversity of underlying databases (de Cock, 1998). The access to multiple distributed databases is managed by software produced in Workpackage 3, which also details the internal structure of the MPS; see Deliverable 3.1 for more information. For the purpose of the current discussion, it suffices to view the MPS plus all related software as a complete module with a single interface, and I will simply call it *Broker*.

3.1 Advanced Access in Decomate II

Workpackage 4, described in this document, conceptually adds three modules to the standard Decomate II system, two of which are clearly marked in Figure 3.1. The third module is the Lexicon (Chapter 5), which is not visible directly (being one of the databases), but which is accessed frequently by both visible modules. In order to maintain clarity, Figure 3.1 does not show the *technical* place of the vari-

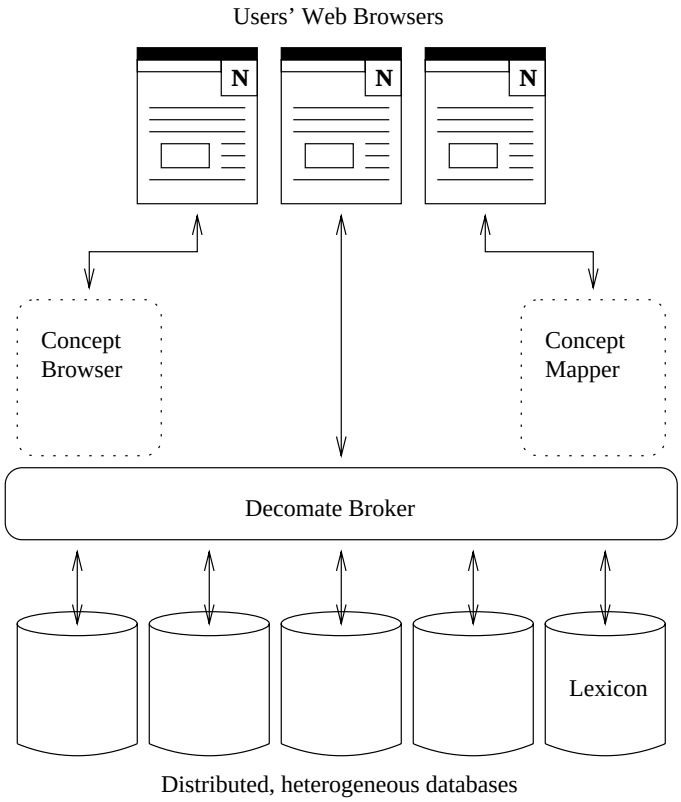


Figure 3.1: Abstract Decomate II Architecture

ous modules. Section 3.2 presents a user view of the Information Space Browser, which appears more integrated than Figure 3.1 suggests.

Instead of doing direct queries to the Broker, a user can select to use the *Concept Browser* (Chapter 4), which will present him or her with a conceptual map of the relevant domain (in Decomate II, Economics). The user can then navigate the domain, while the Concept Browser collects information about chosen paths and nodes. Eventually the Concept Browser generates a normal query which is passed to the Broker.

Broker results can be directly passed back to the user, or be post-processed through the *Concept Mapper* (Chapter 6). This module will be able to augment the query results with semantic information and offers document de-duplication, relevance ranking, relevance feedback (Fitzpatrick and Dent, 1997; Salton and Buckley, 1990), and electronic document tracing. It is our intention to make the Concept Mapper also available for direct queries to the Broker, i.e., without first going through the Concept Browser. However, the amount of extra information available after a Concept Browser session might significantly increase the effectiveness of the Concept Mapper, especially in the area of relevance ranking. Note

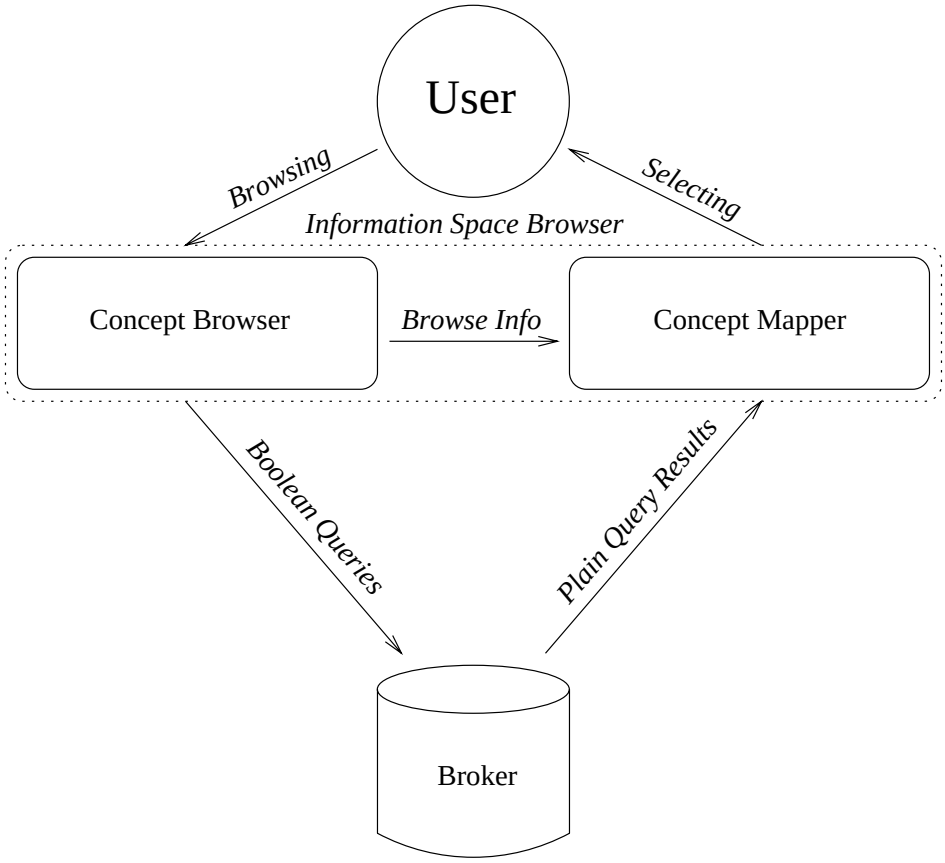


Figure 3.2: Abstract ISB Architecture

that the database engines themselves do not necessarily provide relevance ranking, if any ranking at all, and that the ranking results of separate databases are very difficult to compare and merge (Section 6.3).

3.2 User View of the Advanced Access Modules

Although technically the Concept Browser and Concept Mapper are two separate modules, each with clearly defined tasks, the user will see only the Advanced Information Space Browser (ISB)—actually a combination of the Concept Browser and Concept Mapper in one user interface. This ISB is offered to the user as a normal option in the standard Decomate II interface, and can be selected instead of the plain keyword search interface.

When a user selects the advanced interface, a graphical application[1] will be

[1] Actually, a Java applet running in his/her Web browser.

started that takes over all interaction with the Decomate II system from then on. However, the application will not offer standard Decomate II options such as volume reservation and interaction with the Current Awareness Server. The ISB is intended for information space browsing only. Of course, the user might select to open multiple instances of the Web browser—there is no technical reason why the system could not support an ISB next to a standard session, since the ISB acts as a limited standard session as far as the Broker is concerned.

The 'advanced' way of working in Decomate II is as follows (see Figure 3.2). The Concept Browser part of the ISB presents a part of the semantic field associated with the library (in Decomate II, Economics). By means of the user's initial (traditional) keyword query, the CB makes an educated guess which concepts might be of interest to the user, presents them in the graphical interface, and then surrounds these concepts with related other concepts. The user initially browses through the Information Space (containing only concepts, no terminology or documents), marking specific concepts and linking them up to compile an abstract representation of his/her information need. The number of concepts on the screen, their relative positions, and the concepts themselves will be dynamically adjusted according to user input.

Whenever the user sees fit, usually after a reasonable model of his/her information need has been completed, the Concept Browser converts the conceptual representation into a plain Boolean query that is sent off to the Broker. The Broker (actually the rest of Decomate II) processes this query in the normal way and sends back an unordered list of document index records. De-duplication will likely be done by the back-end as well (see Section 6.1), although the de-duplicator organizationally is part of the Advanced Functionality Workpackage 4.

The unordered list of search results is post-processed by the Concept Mapper using both stateless code and conceptual information coming from the Concept Browser. A combination of a plain result list and a mapping of the results into the already present conceptual model is expected to produce the best feedback as to which documents were matched by which concepts. The user then is offered the possibility to either 'drill down' to a record and view the associated document information,[2] or to explicitly mark this record as 'relevant' or 'irrelevant,' thereby adding information to conceptual model already available in the Concept Browser, a process called *relevance feedback*.

Naturally the conceptual model itself might also be modified by adding or removing concepts. The retrieved document records at least suggest in which ways the user should expand or shrink his/her query. A subsequent Boolean query generated by the Concept Browser will then, hopefully, be more on target. See Section 6.5 for a discussion of Relevance Feedback.

[2]Or the document itself, supposing that the document is electronically available.

It is not decided yet if such a tuned conceptual model, containing both concepts and document records, can be saved for later reuse. If it can be saved, there is the question whether this will be a personal model or a shared model. The basic Lexicon is able to accommodate annotations, and a superconcept such as the final conceptual model might be proposed as a new, higher-order concept. Chapter 5 contains more details about this annotation mechanism.

3.3 Implementation Considerations

For various reasons, explained in the chapters 4, 5, and 6, the ISB will not be built into the Broker but will be uploaded to the user's Web browser in the form of a Java applet. This will give us much more flexibility in the user interface, a requirement for successful network navigation, while significantly offloading the server back-end.

Java applets have the advantage above standard programs that they are architecture-independent and can be maintained centrally. They do not exhibit the installation and maintenance problems usually associated with platform-specific software that must be installed on the PC where the browser runs. Furthermore, they enable remote access to the Decomate II system from every browser that has Java capability, including non-PC platforms such as Unix workstations and dedicated Web browsing machines. Especially in library environments, the zero maintenance 'Network Computers' such as Sun Microsystems' JavaStations most likely will play an important role in the near future.

However, Java is a relatively low-level language, and building a good GUI in Java takes considerable effort. For experimental purposes, we might decide to do a quick prototype implementation in Tcl/Tk. This is a higher-level scripting language which has the same cross-platform capabilities as Java, but which needs to be installed separately as a plug-in (no known browsers come with Tcl/Tk standard, while they all come with Java). When the prototype stabilizes, a re-coding effort in Java should work out much faster.

The distributed architecture with Java applets and Lexicon Server has successfully been used in comparable systems, such as IBMs *Lexical Navigation* (Cooper and Byrd, 1997), and should be flexible and scalable enough for large-scale production work.

3.3.1 Attractiveness

A great deal of effort must be put into an attractive user environment, which makes users sufficiently comfortable to spend some time on query refinement. This is so important because numerous research projects (Lesk, 1998; Jansen et al., 1998;

Fitzpatrick and Dent, 1997) indicate that typical query engine users heavily prefer extremely short queries (1.5 word on average).[3] People are lazy and rarely try feedback on raw queries (*More Like This*) or read the instructions, even if this would give them huge improvements on their query's success. They do not use many Boolean operators either and rarely understand them, and have a tendency to use a limited amount of only 'standard' terms in their queries.

As Jansen et al. conclude, the limited use of advanced search techniques would seem to support the continued research into new types of user interfaces, intelligent user interfaces, or the use of software agents to aid users in a much simplified and transparent manner. This is exactly what Decomate II's Concept Browser attempts. The Concept Mapper (Chapter 6) will attempt to improve usage of relevance feedback in much the same way.

The rest of this document will discuss the Concept Browser, the Lexicon, and the Concept Mapper in greater detail.

[3] These numbers are based on typical Web search engine queries. But since the Decomate II system will be Web-based, it seems prudent to assume that people will treat it as a standard Web query engine.

Chapter 4

The Concept Browser

The Concept Browser (CB) is the main item of the complete Information Space Browser that the end-users will see and work with. Running in the user's Web browser, the CB presents a two-dimensional graphical picture of (a part of) the terminology field associated with the subject the user has selected. As the user browses through the field, the CB dynamically loads in more terminology from a Lexicon Server (see Chapter 5). The end result of a browsing session is a specific query, compiled by the CB, that is sent off to the Broker to be processed. The returned results (bibliographical references) are mapped back into the conceptual network by the Concept Mapper (CM, see Chapter 6), and the user can use this modified map for both document retrieval and relevance feedback.

4.1 Restrictions for Decomate II

The CB architecture as outlined above was inspired by several practical considerations. Unlike many research systems, where a concept browser of some sort is directly driven by the underlying documents from a dedicated, newly constructed database (Cooper and Byrd, 1997), our CB must function in an existing library environment. Several implementation and operational restrictions complicate the addition of concept browsing and mapping modules to Decomate II (Hoppenbrouwers, 1998):

- Wide variation in underlying databases (query language, available information, level of detail, result ranking. . .).
- Need to integrate with existing system components, even when they are not 100% suitable. This holds especially for the various databases, some of which are outside library control (e.g., CD-ROM databases provided by third parties).

- Availability of Boolean keyword queries only; no statistical (e.g., vector, cluster, or frequency) data on the databases is or will become available, and even stemming or truncating options are not guaranteed.
- Restrictions on available system capacity, in particular processing and network bandwidth, with firm limits on allowable response time (taking into account that database queries may already take up lots of time).
- Limited conceptual network creation and maintenance capacity by librarians and documentalists.
- Possibly distributed conceptual networks which should be connected in a federated way, or duplicated and merged.

Especially the given database structure sets this project apart from earlier attempts; we do not have the possibility to create a well-tuned, dedicated database engine with state-of-the-art index and search techniques. Furthermore we emphasize the usage of *purpose-made conceptual networks* for the Browser Interface, while other projects (Cooper and Byrd, 1997) often use machine-generated lexical networks. We believe that it is more likely to get satisfying results by keeping the network in the hands of qualified professionals. In a sense, we do not aim to produce a virtual library, but a virtual *librarian*. Lastly, there should be provisions to partition the conceptual networks to facilitate load balancing, knowledge balancing (specialized networks), and maintenance balancing. Linking conceptual networks in a federated manner is not trivial.

4.2 Design Intentions

What we are going to create is an interface where a user can browse the complete semantic field in an intuitive way, while the system follows his/her traces in order to compile a good query for the actual library database. This involves using both implicit and explicit user signals in order to collect the required knowledge. For example, taking certain ‘turns’ in the network, circling around a concept, and other implicit navigational actions give clues to the underlying ‘knowledge snooper’ what the user might be interested in. Explicit cues, e.g., by marking certain concepts as being of special interest, then provide stronger input to the snooper and can be used to actively propose certain new pathways.

The browser interface in this way assumes almost the role of an *agent*, a semi-intelligent piece of software that plays the role of a *virtual librarian*, guiding the user through the massive amount of knowledge available in the conceptual network (and eventually in the underlying library).

Research has suggested some other ways to enhance users' queries. Interesting developments are history extrapolation through machine learning (Chen, 1995), specifically-linked example documents, interactive term and field suggestions (based on previous queries from other people (Fitzpatrick and Dent, 1997), but reviewed by a documentalist), and direct user-machine dialog to create 'pre-query relevance feedback' (Cooper and Byrd, 1997). It is not likely that the current Decomate II project can use many of these suggestions, so in this report we will not pay attention to them. However, for future enhancements these approaches should be reconsidered.

4.3 Concept Browser Design

This section presents the design such as it can be made at this moment in the project. The inherent innovative approach will likely call for design changes later on, but care has been taken that their impact will not have consequences for any other software module in Decomate II. Especially the communication protocols between the various modules will not need to be changed at all; the Concept Browser acts as a normal Decomate II client to the Broker, using the standard Decomate protocol (HTML).

4.3.1 Initial Positioning

Constructing the initial network to start browsing is difficult, since the expected user input here is the common 1.5-word keyword query (Lesk, 1998; Jansen et al., 1998). With only one or two terms to start with, we need to completely rely on the Lexicon to provide us with a meaningful partial network. Using the standard concept expansion mechanism as outlined in Section 4.3.2, the Concept Browser will call up adjacent concepts from the Lexicon and link them up following the Lexicon-supplied links only. What is displayed is quite comparable to the results acquired by a traditional thesaurus query.

Various ideas to enhance the initial keywords are available, but since the Concept Browser is not at all meant for simple one-shot searches, we will not go beyond the straightforward Lexicon expansion before the other parts of the Concept Browser are firmly in place. After all, the basic idea of the Concept Browser is that the user refines his/her query by browsing—it is not a magic box to 'guess' what the user wants to find.

Care must be taken to not display the actual *term* that the user entered as initial query, but the associated *concept*. This involves lexical (verb, noun, adjective), morphologic (singular, plural) or even syntactic (adjective and noun together) processing of the query. It would be wasted effort to try to include semantic parsing;

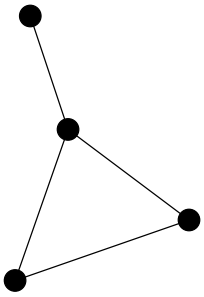


Figure 4.1: Concept Browser Initial Situation

very few users would actually enter syntactically correct phrases, and natural language processing still is not robust enough.

Based on Lexicon contents, concepts closely related to the directly retrieved ones should also be presented to help the user in drawing the border line around the relevant part of the network, see Section 4.3.3. However, these concepts should not be included as positively relevant; their only function is to trigger the user in defining the border of relevancy in the network.

The result of the initial query is displayed as pictured in Figure 4.1. Assume that the center concept is the one triggered by the user query, and that the surrounding concepts are the ones retrieved through Lexicon references. Links are indicated by lines. The figure shows no text because it is not an exact representation of the actual Concept Browser GUI. The final system will present much more information than just nodes and edges, with many configurable options.

4.3.2 Basic Network Layout and Navigation

Given an initial network, either produced by the initial position algorithm (Section 4.3.1) or by the Concept Mapper (Chapter 6), the Concept Browser must produce a two-dimensional layout that visualizes the relationships between the concepts. Since the network itself is highly dynamic, a layout algorithm must be used that can position the concepts on the map in quasi real time. Variants on the 'gravity' or 'magnetic' minimal-entropy algorithms, that drift the concepts into positions where the average distances are as large as possible, seem a good choice.

Figure 4.2 illustrates the principle of gravity drift. From the initial configuration, extra concepts are added on top of the existing concepts depending on the connections. The algorithm then drifts the concepts apart to form a new configuration where node distances, edge lengths, and edge angles are equalized. Generally this leads to a proper distribution of the concepts over the available display area.

Manual manipulations by the user must take precedence over automatic lay-

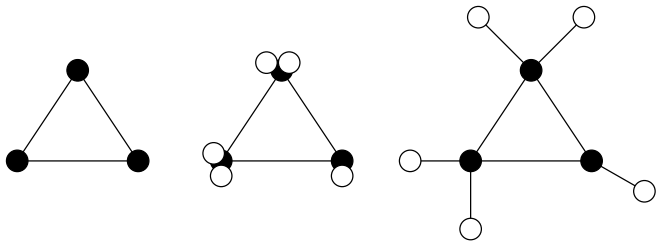


Figure 4.2: Gravity Drift

out; by simply dragging a concept to another place, the automatically layouted concepts must make room for the displaced concept. If we write the 'gravity drift algorithm' in such a way that concepts do not cross links, the generic layout of the network will be maintained unless the user manually displaces concepts.

The layout algorithm must be damped sufficiently to not cause continuous oscillation of concepts between instable positions, an effect that plagued previous implementations. It would be best if the new layout is visually established in one smooth movement from initial to stabilized configuration that takes between two and three seconds, after which the layout engine is shut down to save CPU cycles. In case of performance problems, e.g., with large networks on display, the visual movement of concepts on the screen should be suppressed or the jump distance should be enlarged. For smaller networks, the smooth movement to new positions will help the user to maintain situational awareness. It also is a great visual attractor, well in line with the user attractiveness design guidelines described in Section 3.3.1.

The user can navigate the network by clicking on concepts to expand the network with related concepts, or by deleting concepts. Established user interface guidelines suggest to use a left-click for concept expansion and a right-click to pop up a 'concept menu.' This menu allows further interaction with the concept, such as deleting it from the (browser) network, adding annotations, adding explicit relevant/irrelevant markings, etc. The whole design of the network navigator must be aimed at easy, attractive network browsing and editing. Getting a graphical designer involved later in the project might be worth the effort.

4.3.3 Advanced Network Navigation

Concepts can be presented as either *relevant* or *irrelevant*. When they initially appear in the network, all concepts are assumed to be relevant. Relevant concepts can be marked irrelevant by the user, or they can be completely removed from the network (deleted).

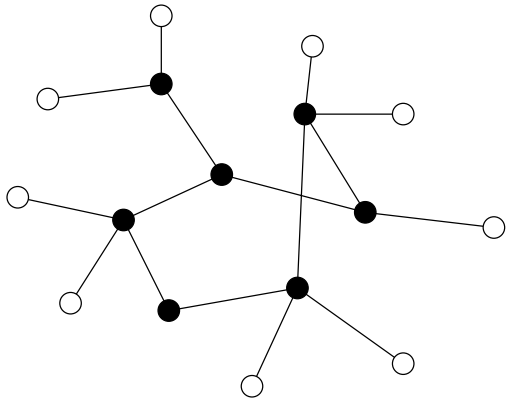


Figure 4.3: Concept Browser after Concept Deletion

There is a distinction between concepts that are not part of the network (invisible) and concepts that are marked irrelevant. Invisible concepts (either deleted or not yet retrieved) are not included in any generated query, while irrelevant concepts are used to create 'not' keywords in the query. This makes irrelevant concepts *important* nonetheless.

A difficulty is that intuitively, deleting a concept probably will be assumed to have the same 'meaning' as marking it as irrelevant. From the user's point of view, irrelevant concepts should *not be* in the network, while for the system, it is essential to *keep* them in the network. It will likely be effective to consider explicitly deleted concepts to be actually marked as irrelevant, and keep a list of deleted concepts internally for this purpose. This would mean that there is a distinction between concepts which are not in the network yet, of which we do not know anything, and concepts which have been removed from the visible network, of which we know that they are irrelevant, yet to the user both of them are not visible.

A compromise would be to lump both deletion and marking irrelevant together in one action, 'delete,' but always keep the concept in the network, but dimmed out to a light shade of gray (Figure 4.3). This way, deleted concepts are clearly set apart from current concepts, yet they still indicate where the user decided to 'draw the border line' in relevancy. The major advantage of this approach is that it now becomes feasible to *visibly* introduce the much-neglected 'not' Boolean operator with almost no effort for the user.

If we encourage users to create a partial network of accepted relevant concepts surrounded by deleted irrelevant concepts,[1] we effectively have a good source for negative keywords and also clearly outline the relevant part of the network—much

[1]It will not always be feasible to keep the deleted concepts on the edge, but the interface will try to move them there.

better than when just stopping at the edge without visible irrelevant concepts. This 'border line' gives a clear reference where the user has explicitly stopped expanding the network, while a group of relevant concepts that suddenly ends leaves him/her wondering 'what else is available over there.'

In case this way of working proves to be too abstract for the user, we can introduce a switch that selects all deleted concepts to be visible or invisible. However, the notion of 'not' is extremely important for the underlying Boolean retrieval engine, and efforts to hide this from the users might turn out less fruitful despite their complaints. The result of the whole operation should be an increase in precision, since the system can compile much better queries if it has the whole range of Boolean operators (AND, OR, and NOT) available.

We are aware of the danger of using 'not' operators. Just as with the 'and' operator, having one 'not' too many can be disastrous for recall while not noticeably improving precision. However, in the proposed system it is not the user who explicitly adds operators and terms—it is the system itself, using pre-determined 'terminology buckets' from the Lexicon. These buckets are carefully designed to match a specific concept, and when such a match requires a selective use of 'and' and 'not' operators next to the 'or,' this should not be dangerous at all. In case of doubt, it will always be possible to decrease the effectiveness of the irrelevant concepts on the final result list. Another method would be to displace 'and' and 'not' filtering to the Concept Mapper, see also Section 4.3.5.

4.3.4 Multiqueries

Aside from the manipulations of the network that increase precision, a user might want to divide the network into several parts that individually specify a single field of interest (a *superconcept*) as tightly as possible, but that together specify several fields. By providing several 'instances' of the same network that can be individually modified, but are linked together at query time, users can build up extensive queries that 'add up' without individual recall and precision degradation.

We selected this option instead of the traditional 'or' link between concepts, because most users do not fully understand the impact of Boolean operators, especially not when multiple phrases and nested parentheses come into play (Lesk, 1998; Jansen et al., 1998; Fitzpatrick and Dent, 1997). The Concept Browser should be used to specify individual superconcepts (networked basic concepts), which then can be stacked up for essentially individual queries.

The Concept Browser interface should provide for easy access to these multiple networks, and basic operations such as partial network copying. Local saving and reloading of networks might be supported in the future; however, it seems better to use the Lexicon annotation function for this (Section 5.3.4), because a well-tuned partial network (superconcept) should be upgraded to a new

concept and made available to other users.

The Concept Mapper (Chapter 6) also has a function that directly influences the GUI; refer to Figure 6.1 on page 46 for more information about this.

4.3.5 Query Compilation

The Query Compiler has the task to find a way from visible network concepts and related concepts not yet visible to a plain Decomate II keyword query. The obvious mappings are from concepts that are called up and retained (they become 'and's) and those that are called up and rejected (they become 'not's). Having only 'and' and 'not' clauses in the Boolean query means that the precision could be alright, but several things must be considered to prevent huge loss of recall.

The main instrument to improve recall is the Boolean 'or,' which lets the Decomate II back-end select multiple matches. The user interface gives explicit cues for 'and' and 'not' clauses; the 'or' clauses, however, have to be implicitly added without user intervention.[2] This is made possible because the Lexicon is a *conceptual* database and not a terminology database or thesaurus.

Each concept is attached to several terms (words), which together share a well-confined meaning. Ideally, the terms under one concept should be complete synonyms—they should be able to replace each other in all linguistic situations. In practice, a somewhat less restricted selection can be made, since we are not building a translation device where minor linguistic features are all-important. The WordNet lexical database (Miller et al., 1993) contains numerous examples of 'synonym sets' that serve as good suggestions for how large a concept can become.

The concepts are linked together by semantic links, such as 'broader term,' 'narrower term,' and other meaningful connectors described in Chapter 5. Some of these links will be useful in attaching some more concepts to the query if the user has not explicitly denied their relevance already. Such additional concepts might be positive ('or') or negative ('not') depending on the link type. Situations can occur where additional concepts as a group are 'or'-ed, while the group is 'and'-ed within itself to exclude synonymous words.

Even when the Lexicon provides the Concept Browser with an abstract view on the terminology (the concept/term separation), simply using the plain underlying terms of a concept will not work. Care must be taken that various morphological forms of the same concept will exist in the document index records, such as singular and plural, the various inflections of nouns, and the many forms verbs can

[2] Note that a user does specify 'or's when creating multiple superconcepts, but these individual concept networks are considered separate queries.

appear in. Although it is possible to build a generative Lexicon that contains all of these morphological variants of the words (Hoppenbrouwers, 1997), using the full expansion in the query would result in up to fifteen forms of the same word 'or'-ed together—and that is still only one word. With synonyms and related concepts, queries of over one hundred separate items would occur, and it is not at all certain that the Decomate II back-end could cope with this.

Traditional approaches solve this problem by using *stemming algorithms*, that essentially truncate words so that the word endings (which usually contain the bulk of the morphological variance) are removed. Some Decomate II databases support truncating (with the $*$ operator), but many do not. It is not yet clear how we are going to work around this limitation. A possibility might be to include carefully selected morphological variants only, such as plural and singular for nouns, but restrict the verb inflections. A helpful factor is that the mainstay of Economic literature is in English, a language with a rather sparse morphology.

Externalizing Recall Limiters

The precision-improving operators 'and' and 'not' can have a disastrous effect on recall when they are used too often and with the wrong terms. Instead of the hard Boolean rules, a somewhat relaxed rule might be more useful here, i.e., irrelevant concepts do not completely shut out documents but merely cause a lower ranking on the results list.

For this approach to work, the actual database (Broker) query should not be limited at all (should only contain 'or' operators), while the Concept Mapper takes care of the subsequent relevance ranking using the 'soft and' and 'soft not' operators on the records itself. Obviously this will be an expensive approach if the database back-end returns several hundred or thousand records because the 'or' query gets out of hand.

We propose to use explicit 'and' and 'not' terms contained in the Lexicon terminology buckets directly on the database back-end, because these operators are explicitly added by Lexicon maintenance people. The user-marked irrelevant concepts then carry their terminology buckets over to the Concept Mapper, where they are used to adjust the document index record relevance ranking. In this way, we keep tight control over the way in which the dangerous operators influence the result, while not sacrificing any form of recall limiting.

4.3.6 Network Distribution

A large part of the Concept Browser's usefulness is in its ability to combine several Lexicons into one single interface. This opens up the possibility to maintain

distinct Lexicons for distinct semantic fields, such as Economics and Information Technology, which can be linked up afterwards without explicit cues being put in by the maintainers. Naturally, linking two Economics fields from different universities would also be possible.

The Lexicons themselves have no specific functionality to make this distributed conceptual network possible. Each Lexicon is an independent collection of concepts and terminology, although it would be technically feasible to store several Lexicons on the same Lexicon Server. The complete merging of Lexicons is done on demand by the Concept Browser, running in the user's Web browser.

When a person using the Concept Browser selects more than one Lexicon as a source for the conceptual network, the CB will use the underlying terminology (actual words) to 'guess' which concepts in both Lexicons might overlap. It will be impossible to *guarantee* that the found concepts indeed are synonymous, but the more terminology overlaps, the higher the probability that the match is valid. More advanced methods of mapping will use the available semantic links between concepts, to see if concepts that are expected to overlap for terminology reasons also have the same basic type of connections to other concepts. One step further then, obviously, is to compare adjacent concepts on terminology to get a further confirmation of overlap.

This unregulated way of Lexicon matching, although fuzzy, has one big advantage: Lexicons that have not been built with matching in mind can still be linked up, providing interesting comparisons between semantic fields that are traditionally unrelated, such as with Economics and Information Technology that meet in Electronic Commerce.

However, if certain core concepts emerge in various Lexicons which describe the same generic field, it will be advantageous to standardize some concepts and give them a unique identification code. This might be a word, but with the specific connotation that this is *no simple term*; it functions only as a mnemonic identifier for a more complex concept. If Lexicons have enough of these universally accepted and identified concepts available, a situation that is not feasible in the near future, matching them would be greatly simplified. The same is true for cross-lingual Lexicon matching, although this problem can also be tackled by putting multi-lingual terms in the same concept within one Lexicon. In this way, a single multi-lingual Lexicon can support several monolingual Lexicons to 'find' each other.

Various ways of presentation can be used, ranging from a completely invisible merging of all available Lexicons to giving each concept a distinct color depending on its Lexicon of origin. Experiments will have to make clear what method, if any, will be most effecting in library practice.

Chapter 5

The Lexicon

Central in the Advanced Access module, although not directly visible to the users, is the Lexicon. This is a database that contains a part of the conceptual model of the relevant scientific field (in Decomate II, Economics), both the semantic structure and the terminology. The Concept Browser accesses the Lexicon in real-time every time a new concept is required.

The Lexicon has two separate aspects that both must be dealt with: the contents aspect and the system aspect. Comparable to a database system, a Lexicon has an underlying engine (the Lexicon Management System), a schema, and data. For our purpose, the engine and the schema can be considered static and together form the Lexicon Platform, while the data is formed by the conceptual and terminological contents. See also (Hoppenbrouwers, 1997).

In order to define the conceptual model and restrict the working definition of the Lexicon, we present several definitions such as used in various scientific fields (computer science, artificial intelligence, library science, linguistics) and select the appropriate one for our purpose.

5.1 Project Terminology

There are several descriptions and definitions of conceptual networks, sometimes called *ontologies*, *thesauri*, *lexicons*, or *semantic fields* (Hoppenbrouwers, 1997). Traditionally the difference between these and other related terms is the following.

The vocabulary provides the official list of correct forms of words, presents only syntactical features, and gives idiomatic patterns of usage if necessary (Weigand, 1990, p.77).

A thesaurus provides the official survey of correct terminology for concepts, presents only basic semantic features, structures the terms in a semantic

net, and adds to the vocabulary special patterns of usage appropriate to the special concepts (Weigand, 1990, p.77). Alternatively, a thesaurus is the vocabulary of a controlled indexing language, formally organized so that *a priori* relationships between concepts are made explicit (Aitchison and Gilchrist, 1987).

A dictionary presents the definitions of terms from the thesaurus, gives humans the understanding of specialized words, helps shape the growth of the thesaurus, and helps authorities in deciding on the admission of new terms (Weigand, 1990, p.77).

An ontology is 'a systematic account of Existence,' a description of the minimal set of concepts that a language needs to express all its other concepts (Kaminsky, 1969). Pragmatically, an ontology defines the vocabulary with which queries and assertions are exchanged among agents (Gruber, 1993).

A lexicon combines the vocabulary and the thesaurus and integrates them in a machine-readable format so that it can be managed and queried by computation engines (Hoppenbrouwers, 1997).

A semantic field or *domain* is a more or less clearly outlined subsection of the real world. It can often be related to a group of people who live and work in the same environment, a *semantic community* (Robinson and Bannon, 1991; Ulijn and Strother, 1995). Some authors claim that within any semantic field, there must be no ambiguous terminology (Wiederhold, 1995), thereby putting a very hard constraint on the semantic field up to the point that only one single person can live in such a field at the same time.

A conceptual network is a collection of semantic nodes with links between them, in such a way that many relationships are captured. Covering a semantic field, it is usually much more extensive than a typical thesaurus, e.g., containing semantic roles and part-of relationships (Miller et al., 1993). However, newer thesauri contain more and more information and can be assumed to be conceptual networks as well (Miller, 1997).

Throughout this document we use the term 'Conceptual Network' to refer to the abstract network of concepts that is used to describe the semantic field of Economics, and 'Lexicon' for the actual implementation of this conceptual network in data. The machine containing this data and its management software is called 'Lexicon Server.' Since concepts usually have more than one associated term, the Lexicon also contains the actual vocabulary of the semantic field, plus the part of the thesaurus that defines preferred terminology.

5.2 Lexicon Content

The Lexicon is a highly dynamic database, not comparable to the traditional semi-fixed classification schemes that have been used to index documents for decennia. Not only will the Lexicon evolve over time more rapidly, it also contains much more information, both on the conceptual level (relationships) and on the lexical level (actual terminology, spanning multiple scientific fields and languages).

5.2.1 Structure

Most 'strict' ontologies or thesauri that propose one single hierarchical organization of terminology are not sufficient to serve as a conceptual network (Sowa, 1983, p.15); no linguistic or psychological evidence has uncovered a truly universal set of primitives, and often it is difficult if not impossible to assign concepts to only one category (Woods, 1997). Likewise, because the vocabulary of each living language grows with approximately 6000 lemmas per year, especially in the technical-scientific register (Ulijn and Strother, 1995, p.101), it will be very hard to claim that *any* thesaurus is ever complete. Regular updates must be applied to a thesaurus to keep it in synchronization with the evolving semantic field (Aitchison and Gilchrist, 1987).

The dynamic nature of thesauri and conceptual networks means that mostly static, hierarchically organized classifications such as the UDC tree[1] or the classification of the Journal of Economic Literature (JEL)[2] are not sufficient to serve as a complete conceptual network. Besides, they do not aim at covering the terminology of the semantic field—they want to identify specific subfields (subjects) within the larger fields. Of course their *subject headings* can be used as a starting point for thesaurus construction, and they can be included as generic 'see also' pointers in a conceptual network.

Summarizing, a true conceptual network must be organized as, indeed, a *network*, with multiple parents per node and a substantial number of link types capturing as many relevant semantics as possible.[3] For examples of link types and a generic theoretical overview of Lexicon organization, see (Hoppenbrouwers, 1997).

5.2.2 Re-using existing resources

Conceptual networks such as WordNet (Miller et al., 1993) contain enough terminology and relationship information to be usable, however, they usually are too

[1] `http://main.bib.uia.ac.be/MAN/UDC/udce.html`
[2] `http://www.econlit.org/elclasbk.htm`
[3] Unlike, for example, Verity TOPIC's meaningless link weights.

static as well and cover a broad range of common semantic fields while being sparse on detailed, specialist fields—which are far better suited to assist users in knowledge navigation (Bodner and Song, 1996; Howard, 1992). It is especially important to have the conceptual network organized in terms of, indeed, *concepts*, instead of plain terms. WordNet uses the *synset* primitive to group highly synonymous terms together, and the EuroWordNet Project extends the synonymity relation to include multiple languages (Vossen, 1997; Vossen et al., 1997). Other work on Lexicons, aimed specifically at conceptual modeling (Hoppenbrouwers, 1997), also suggests ways of organizing terminology to properly present a conceptual space to users.

Acquiring a suitable conceptual network therefore is not just a matter of copying existing thesauri or term lists. Considerable effort should be put into the initial creation of a conceptual network for knowledge navigation purposes. This is not to say that existing information cannot be reused, but it usually requires extensive post-processing.

Besides the acquisition problem, there is also a maintenance problem, a navigation problem, and a mapping problem. The next sections will consider each problem in more detail, suggesting possible solutions as appropriate.

5.2.3 Maintenance

Any semantic network which models a piece of the world needs regular updating in order to stay synchronized with the world. The idea that a network could be constructed once and remain stable for an extended period of time should be abandoned:

> The danger is that if the thesaurus is permitted to become monolithic and resistant to change, it can actually hinder both indexing and retrieval.
>
> (Milstead, 1992)

There are two separate groups of people who naturally should get involved in conceptual network construction and maintenance: library professionals (documentalists and librarians) and research professionals (scientists) who regularly use the library.

The Role of the Documentalist

In case of a library system which specializes in one particular scientific field, such as Economics, the network should be maintained by experienced documentalists

who are comfortable with this field. These people can quickly recognize the particular spots in the network where potential new concepts should be placed, and can update and use the network as part of their regular work. In this way they develop a 'map' of their field, which can be very useful for other purposes besides knowledge navigation support. We assume here that documentalists are explicitly keeping up with the scientific field; not that they just catalog new publications. Only then, proper maintenance of the conceptual network is guaranteed.

Of course, it should be assured that network maintenance is a technically simple operation that does not need much training or time, and that the immediate day-to-day advantage for the documentalists is sufficiently clear. Only then will these people be inclined to spend effort on network maintenance. This is an important part of the whole project, because without network maintenance, the relevance of the concept browser will diminish over time.

Automated tools should be available to help documentalists with network maintenance as much as possible, e.g., by proposing specific places in the existing network for new concepts. The documentalist then might only need to confirm the system's proposal. In addition, tools for network manipulation (moving, copying, deleting, printing) should be readily available. A good browser which exceeds the simple record-oriented concept view (a single concept with all associated direct links only) is also required, in order to facilitate situational awareness.

The Role of the Scientist

It is a reasonable assumption that the same scientists who use the library have at least a partial task in providing its contents as well. Not only do they influence the collection (although usually only in an indirect way), they also publish documents about the same scientific field. This implies an intimate knowledge of at least a part of the field.

Leaving all the network maintenance to documentalists and librarians denies the obvious knowledge that some library users already have. Especially during the initial network construction, library people should consult researchers in order to build up a coherent and reasonably complete network. Note that it is not required that all scientists fully agree on the network; it is no standard classification. The purpose of the network is to assist less experienced people, and when several 'research schools' exist, that fact should be noted, not voted away.

But also when the system is in production, scientists (users in general) who browse the system might have valuable suggestions for Lexicon changes and additions. When these suggestions must be filed through a forms interface or by E-mail, it is almost guaranteed that not much feedback will be generated. It would be preferable to have a very simple 'annotate' option right in the Concept Browser, that gives users the opportunity to immediately add comments and suggestions to

existing concepts, and also to propose completely new concepts with some initial links. These user additions should be shown in the browsers of all users, but with a different presentation (e.g., color), and with the option to switch them off. The official Lexicon maintenance people then should be able to review the suggestions and to accept or reject them with the click of a button. Alerting mechanisms should be in place to prevent the maintenance people from having to exhaustively search the Conceptual Network for user suggestions.

5.2.4 Navigation

The networks (thesauri) typically used in information retrieval systems are standard more generic/more specific hierarchical trees, sometimes extended with cross links such as 'used for' and 'related terms.' Synonyms may be present, but the main purpose of classification hierarchies usually is to avoid any synonymous references.

Obvious links such as between 'software engineering' and 'software protection' are often missing, even when both terms are present, because they do not typically classify under the same head word. As an example, in the thesaurus used by Excerpta Informatica at Tilburg University, 'software protection' was classified directly under 'software,' but 'software engineering' was generalized by 'software technology,' which was not at all mentioned under 'software' but was located under 'computer technology.' In other words, there was no navigational path between the terms, even though they both started with the same word. Because the Excerpta thesaurus could also be queried as if all terms together were a full text database, the 'software' keyword in 'software engineering' was found, but the result of this query was an alphabetically ordered list of 61 items which did not all contain the 'software' search term.

It is a well-known problem of any hierarchically organized system (even when cross links are present) that concepts often do not naturally classify under one single category (Woods, 1997). Concepts should be placed in multiple locations in the network, participating in several tree structures if required. However, when users navigate a particular tree, it should be made clear to them when they are about to leave their original tree. This is a classical Hypertext navigation problem, and indeed, navigating a conceptual network has many parallels in navigating a Hypertext.

When fully associative Lexicons are navigated through a suitable interface, such as the various WordNet interfaces (Miller et al., 1993), link types such as more generic/more specific are usually made explicit. Other variants exist that use one single link type for all links, but give these links different 'weights' to

indicate the 'strength' of the link between two concepts.[4]

The whole issue of hierarchical navigation boils down to the difference between structural (analytical) browsing and associative browsing. Despite the obvious maintenance and implementation advantages of strict hierarchical concept trees with explicit cross links outside the tree structure, an associative structure is better suited to model a typical semantic field. Psycholinguistically inspired Lexicons such as WordNet therefore offer more link types, like synonyms, antonyms, meronymy, and others (Miller et al., 1993). Presenting these links to the user in an easy format is not trivial; most likely, some form of two-dimensional graphical browser is needed. Considerable work has been done in this area, see Aitchison (1987) and Lancaster (1986).

Many helpful ideas have come from the Lexical Navigation Project (Cooper and Byrd, 1997), where a browser client was developed to facilitate user browsing of a lexical network. Although there are differences between an lexical and a conceptual network, these differences are not necessarily visible to the user. The local, non-persistent network management that can be done, such as moving nodes on the screen or deleting nodes to create a specialized view or subnet, seems very helpful to browsing and query creation.

5.2.5 Mapping

Some current thesaurus-assisted information retrieval systems use the thesaurus to let the user navigate to a certain term, and subsequently use only this term to query the (full-text) database. Although this method certainly helps to suggest particular search terms to the user, it does not at all use the semantic network formed by the links between words to improve the query, i.e., to increase the system's recall and precision.

The semantic network actually serves two distinct purposes. First, it helps the user to become familiar with a certain semantic field which (s)he might not fully grasp yet. In this way, the network might help the user to come up with relevant terms, which will yield better results than forcing the user to key in keywords from the top of his/her head. Second, the network offers the system the option of adjusting the user query by not using only the term the user indicated, but also using the terms around the user term and maybe even other concepts.

The Lexicon contains actually *two* layers. The top layer is the aforementioned semantic network, holding concepts only. The bottom layer holds the actual vocabulary that will be used for database back-end queries. From a simplified point of view, each concept refers to a 'terminology bucket' that contains all synonyms of the concept, comparable to the traditional thesaurus relationship of 'use' and

[4] `http://www.plumbdesign.com/projects/thesaurus.html`

'used for.' However, the concept itself is *not* a member of the vocabulary, it is only a conceptual reference that can have several labels to recognize it, but none of these labels is directly used for database queries.

The 'terminology bucket' may contain multiple language vocabulary, and also extra flags for preferred terminology for certain databases (especially if they are indexed using restricted vocabulary). The query mapper should use this extra information, if available, to increase the quality of query results. It might be a problem getting all this information to the Broker through the standard interface though. We should take care not to implement direct access to the various databases, bypassing the Broker's multi-query.

Besides (near) synonyms, the 'terminology bucket' can also contain more complex items, such as explicit 'not' clauses or explicit 'and's. The idea is that selection of a single concept should invoke a well-tuned query that has been optimized for that concept, and that the user manipulates the semantic network to combine these (sub)queries into one large query.

Candidate thesaurus terms to be called in by the query generator are synonyms (although care must be taken not to expand the query beyond what the user intends), direct hypernyms, some levels of hyponyms, and maybe some levels of meronymic (part-of) relationships (Maulding, 1991). Note that these are *candidate* terms, not necessarily actual terms used for query expansion. Especially synonyms usually pose problems, as the following quote from Woods explains:

> A common approach to the paraphrase problem is to use tables of synonyms to automatically expand queries by adding terms that are recorded as 'synonymous.' However, there are few real synonyms in English, so the common practice is to include related words as if they were synonyms. However, treating terms this way when they are not really synonyms introduces a level of granularity that trades off precision for recall. There is no a priori correct level for this tradeoff—different information needs require different levels of generality—so this technique often degrades retrieval rather than improving it.
>
> As an alternative to synonym classes, we use taxonomic subsumption algorithms that exploit generality (subsumption) rather than synonymy to connect terms in queries with passages that contain more specific terms as well as the requested terms. These algorithms do not automatically explore more general terms, so the level of generality is controlled by your choice of query terms. For example, if you ask for 'motor vehicles' you would get trucks, buses, cars, etc., but if you ask for 'automobiles' you would get cars and taxicabs, but not trucks and buses.
>
> (Woods, 1997)

The use of subsumption to expand a query can be enabled if the Lexicon contains the appropriate 'narrower term' relationships. For most existing thesauri, this already is the case.

More than the above terms (the selected concept plus synonyms and subsumptions) are not available when the user came to a term without any navigation, i.e., by hitting a spot on the 'map' and not moving from there. When the user has navigated the map, however, much more information is available to construct a query.

For example, a common way to end up at a term is to follow one of the available tree structures down from the top. There certainly is a noticeable difference in arriving at 'software engineering' via 'technology' and 'engineering' than arriving via 'computer programming,' 'modular software,' 'component reuse' or even through an associative side link as in 'organization,' 'business process redesign,' 'business process re-engineering,' 'engineering.' Traditional thesauri go to great efforts to prohibit multiple paths to concepts, but a well-designed network might provide invaluable extra clues to determine the nuances and in which context the final term should be interpreted (Aslandogan et al., 1997).

The ability for the user to mark terms while passing them by, so that the resulting query will take the marked words into account as well, can also enhance the standard 'query for this term' feature.

For more details about the way in which we intend to implement the various mappings, see Section 4.3.

5.3 Lexicon Platform Design

A feasible Lexicon Platform (Lexicon Management System (LMS) plus Lexicon Schema) must have the following features to be usable in the context of the Decomate II Project:

Fast Speed is mandatory. A hundred users must be able to access the LMS at the same time without noticeable performance degradation, since for most moves in the Concept Broker, some Lexicon activity is required.

Server-based The LMS must be built as a server, ready to respond to requests over the network. It is not technically required to make this server a full Decomate II Broker-compliant database, and for performance reasons it might be better to skip the Broker. However, the expected Broker performance will be sufficient for our purpose, so we will initially implement the LMS as if it was a normal Decomate database.

Simple Both the server and the communication protocol, and also the Lexicon Schema, must be simple enough to be configured and managed by the same people as the rest of Decomate II. In particular, no bleeding edge linguistic theory, nonstandard programming language (e.g., Prolog), or database (e.g., a true OO database) should be used.

In short, despite the aim of the Advanced Access module, the LMS itself should be as low-tech as possible. The actual innovative character of the module lies in the *combination* of several techniques and pieces of software, and in the Lexicon contents. We expect to be able to use existing and well-known library database technology for the Lexicon.

5.3.1 Performance Considerations

The main activity on the Lexicon is of a simple question/answer kind, typically requesting information about one single concept at a time. Concept access is through concept ID, a uniquely identifying property, most likely an integer. Retrieving the corresponding concept information from the Lexicon database therefore can be done through very efficient algorithms and data structures. Only an initial query using the terminology part of the Lexicon will require an alphabetical search.

In order to prevent all Lexicon data to be uploaded to the client at client startup time,[5] partial Lexicon content delivery is mandatory, hence the server concept. The IBM Lexical Navigation project (Cooper and Byrd, 1997) has successfully pioneered this approach.

TREVI (Esprit EP23311) taught us not to use a commercial OO database management system for a Lexicon. These systems offer plenty of flexibility and a natural organization of the Lexicon, but this at the price of sluggish performance. Given the fact that the Lexicon will be queried orders of magnitude more often than that it will be updated, retrieval performance is more important than insertion performance. The standard database management considerations such as rollback and protected transactions will not be used significantly, and not purchasing a commercial database per Decomate II installation obviously saves lots of money.

Where TREVI used Java as programming language for reasons of marketing and maintainability, we strongly advice against using Java for a performance-critical module such as the Lexicon. Since the LMS is completely server-based, the byte code interpreter would not offer any significant advantages,[6] and none of

[5] Remember that the client is downloaded over the Internet. Current typical Internet bandwidth restricts the maximum applet size to about 50 Kbyte. The Lexicon will never fit into this straightjacket.

[6] Naturally the client front-end will greatly benefit from the byte code interpreter.

the other Decomate II modules currently use Java.

The wish to reuse existing technology where possible leads us to choose a variant of a thesaurus database, based on Z39.50. The Elise II project (LB-4005/A) implemented the AAT thesaurus of the Getty Institute, containing over 20,000 preferred terms plus associated non-preferred terms, using the Zebra index and retrieval engine. This thesaurus has the required two different access paths (on concept and on concept ID), plus the complete set of search features including truncation and regular expressions. Furthermore, Zebra databases are in regular use by all currently active Decomate partners and thus well-known. Their performance is excellent (the Nordic countries use Zebra for their country-wide Web indexing), and Zebra is free for non-commercial use.

Using an open standard such as Z39.50 also allows for relatively easy integration of existing thesauri such as the JEL classification, which often follow the same standards.

Zebra is no 'true' database in the maintenance sense—it is more a search engine. Experiences at Tilburg University suggest that the actual Lexicon database that is maintained could best be stored in some form of relational (SQL) database to suppress as much redundancy as possible. At regular intervals this database can be converted into plain ASCII files that are indexed by Zebra to produce the production Lexicon server(s). An additional advantage of this setup is that it becomes almost trivial to generate special-purpose Lexicons out of the database, e.g., hardcopy documentation.

5.3.2 LMS Architecture

The LMS can be divided into four separate modules (see Figure 5.1):

1. Broker gate keeper
2. Lexicon Server (possibly several in parallel)
3. Lexicon Database
4. Lexicon Maintenance Interface plus Annotations

An end-user client (a Web browser such as Netscape or Internet Explorer, capable of running Java applets) connects to a Lexicon Server through the Broker in the usual way. Note that it is possible to start several servers in parallel on different machines in order to distribute the load; the Broker can use a round-robin algorithm or a smart load balancer to maintain an evenly distributed Lexicon server load if required.

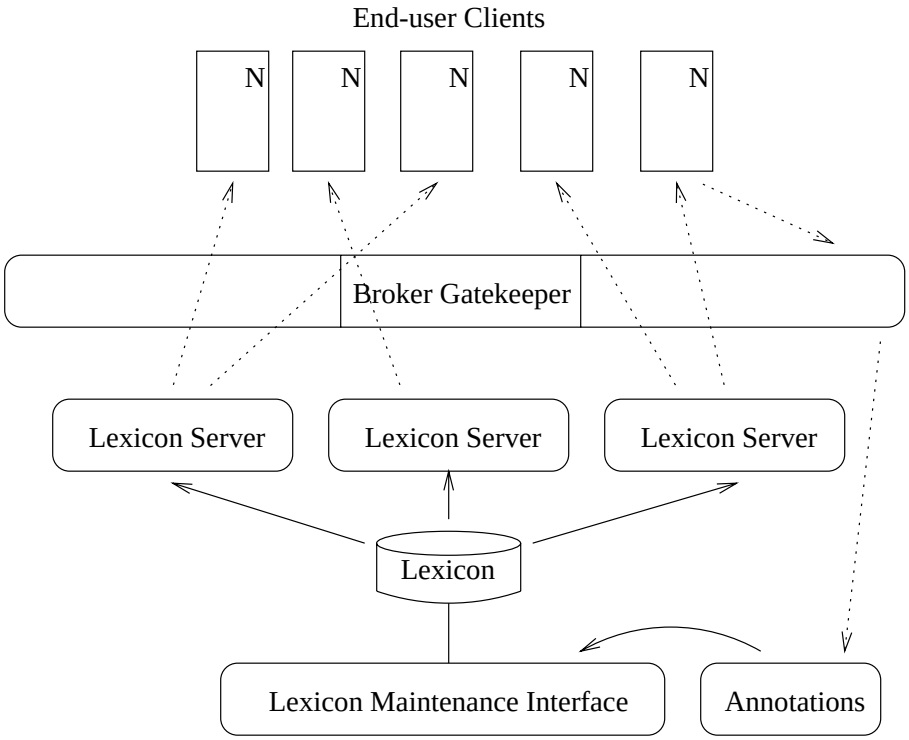


Figure 5.1: The LMS Architecture

The Lexicon Servers all share the same lexicon database (Lexicon), and do not update this database in any way, thereby maintaining the safe (not crash-sensitive, parallellizable) production model. We expect that the actual Lexicon Servers will run on a *local copy* of the database, such as an ASCII file indexed by Zebra. Only the Lexicon Maintenance Interface (LMI, see Section 5.3.3) is capable of Lexicon updates.

An extension of the Concept Browser offers users the possibility to add annotations or suggestions for new concepts to the Lexicon. It currently seems best to channel these update requests via the Broker to a reduced variant of the LMI, that can add the user's input to a separate annotation database. The full-blown LMI then can be used to selectively import the annotation database under guidance of a certified Lexicon maintainer (documentalist).

To facilitate Lexicon sharing and cooperation, features will be offered to incorporate other Lexicons into the database by explicit import through the LMI (invisible to the end-user clients). In special cases it would be advantageous to link up separate Lexicons without merging them into one. This should be done outside the architecture pictured in Figure 5.1, by means of a direct connection between the end-user client and the remote Lexicon Server. Naturally, the Bro-

ker would coordinate such a request (Decomate II Brokers know of each other's existence through their meta-database). Section 4.3.6 expands on this Federated Lexicon approach.

5.3.3 The Lexicon Maintenance Interface

Although at this point in time a detailed design of the LMI is not feasible because the Lexicon internal structure needs to be experimentally reviewed, several features of the LMI are likely to be required.

The main purpose of the LMI is *to help professional Lexicon maintainers with their job*, i.e., it is no library end-user tool. Visual attractiveness and intuitive use are less important here than efficient and effective Lexicon update facilities. The LMI is the strategic tool with which the Lexicon is kept up-to-date, and if this tool is not considered adequate by the documentalists responsible for Lexicon maintenance, the performance of the whole system will suffer due to an inadequate Lexicon.

It is not required to build such a tool as a Web browser-based Java applet or form interface, although it might be feasible. A single application, preferably cross-platform in a language such as Tcl/Tk (Ousterhout, 1994), might be easier to construct and maintain. A separate application might also offer advantages due to its excellent interface possibilities with Unix text utilities, and relational databases. The prototype of the LMI most likely will be an independent application or set of utilities, with a full-fledged graphical implementation becoming available later in the project.

Although availability of a good LMI is crucial for the viability of the Lexicon in the long run, it is no priority at the current time because without a functioning Concept Browser and Lexicon Server, the LMI serves no purpose anyway. Given the possibility to create a Lexicon with standard ASCII utilities, we postpone the actual design and development of the LMI to a later phase of the project.

5.3.4 Annotations

Users of the Concept Browser, either students, scientific faculty, or librarians, must have the possibility to add *ad hoc* annotations to the network as they see fit. A large part of the network's usefulness is determined by the network coverage of the semantic field, and it would not be smart to ignore user contributions.

Naturally, annotations should not find their way into the officially certified Lexicon by themselves. All annotations should be reviewed and acknowledged by the Lexicon maintainers before they are moved from the annotation database into the Lexicon production database. But it should be possible for users to select

if they want or do not want to see these annotations in their Concept Browser, possibly in several stages.

The most likely annotations are *additions* of concepts, terms within concepts, and links between concepts. Occasionally somebody will want to *remove* something which in his/her opinion decreases the effectiveness of the Lexicon in putting good Boolean queries together. *Changes* are combinations of removals and additions.

Other forms are *verbose annotations*, i.e., plain texts to explain or suggest something, and proposals for *direct document references* from certain concepts as 'typical' documents. The last category is both powerful and dangerous. It can certainly help people who are new to a field to get immediate references to a standard document that would have been suggested to them by a documentalist as initial reading, and it is a good help in relevance feedback. Having qualified document references available even before the Boolean query and result processing has been done is a huge advantage for the query compiler. But the danger is that users will focus too much on standard, older documents instead of using the system to retrieve newer and other relevant documents. Moreover, it would make the Lexicon dependent on a given library collection.

An alternative worth considering is to use user feedback on result sets to have a background store of 'good' and 'bad' matches. When users mark some retrieved document references as (ir)relevant, the interface can forward the index records of these documents to the Annotation Database where they are stored for future processing. Review of the records and the associated concepts and terms, either manually or automatically, might suggest to modify the term base in order to better match user expectations in the future.

The net result of annotations is that the Lexicon gets better tuned to what people want instead of to what scientists/documentalists *think* that people want. This is a very relevant feature. The whole Information Space Browser should not aim at absolute 100% accuracy and to cover all possible search cases, but at a generic improvement of a large part of the user queries. Expanding or shrinking certain concept areas to what the common user seems to expect is a powerful tool to reach this goal.

5.3.5 Implementation

Experience with previous Lexicon implementations makes us choose for a single program (Unix process) per Lexicon Server, that does not *fork()* when new connections to the server are initiated. Forking would mean making a full copy of the main memory Lexicon database, which clearly is an undesirable situation. The selected Zebra index engine follows this model.

Connection Management

Two methods exist for providing multi-client access to the same server process: multi-threading and serial serving.

Multi-thread servers use a single thread for each connection, where all threads together share all resources of the process. Since the Lexicon database is strictly read-only and since all connections are stateless (clients require information on a concept, get it back, and that is it), the inter-thread communication will be almost nil; no synchronization between threads will be required. The scheduling of the individual threads will be taken care of by the operating system, and on symmetric multiprocessor machines with shared memory the server performance will scale up nicely.

Serial servers have a single process with a single thread, that serves all clients in a round-robin fashion. New connections are put in a connection table and some semaphore mechanism is used to decide which connection should be served at which moment. During the service of a client, all other clients must wait. Adding more CPUs to the server machine has no effect on the performance of this single process. Starting more processes to use the CPUs is possible at the expense of memory requirements.

Both approaches have advantages and disadvantages. Since Zebra is out of our control, its performance on many small queries should be investigated during test runs of the Lexicon. Fortunately the (un)availability of a multi-threaded Zebra server will have no influence on the clients; either way the connection protocol will be exactly the same. It is feasible to change the connection handler later in the project without changing either the clients or the Lexicon core code.

Connections can and will be interrupted frequently by Internet problems, and the connection manager should be robust enough to handle these interruptions. Furthermore, the very nature of the Web, with stateless network links, encourages clients to ‘go away’ without signing off. A good timeout mechanism should clean up abandoned connections in the Broker, but this is being taken care of in Workpackage 3 anyway.

Query Management

The actual server implementation is quite straightforward. Two types of queries will be used on the server: term queries and concept queries.

Term Queries are typically issued at the start of a Concept Browser session only. The user has typed in a keyword query, the infamous 1.5-word naive document selector, and the Lexicon server must return a set of concepts that closely matches the query.

Concept Queries form the mainstay of the Lexicon Server's job. Concept Browsers request information on one or several concepts, and the server must return this. Concepts can be individually addressed in a unique way through a concept code, and no processing is required.

Term queries can afford to take a bit of time, up to several seconds, since users will expect the system to 'search' for a while. Concept queries, however, should be answered in almost zero time since they are used while the user is clicking his/her way through the network. Users will be severely irritated by sluggish response during navigation tasks.

When merging remote Lexicons, time delays in concept retrieving are inevitable. Solutions can be found in asynchronous processing, with remote concepts popping up after they are retrieved while the user can still click ahead, and in pre-fetching where the Concept Browser fetches adjacent concepts ahead of time in an attempt to have them available when the user eventually selects them.

Protocol

The Lexicon Server will be accessed through a standard Z39.50 protocol over a standard TCP/IP socket. All applicable programming languages and environments can handle sockets, and the stable Z39.50 protocol can be implemented with simple Perl, Tcl, or Java programs.

Data protection and access control are not really relevant. Since the Lexicon Server is read-only and the annotations do not directly influence the main Lexicon database, basic request sanity checking and possibly IP-address checking for annotations should be sufficient. The protocol does not need to be encrypted or binary. Checksum protection over a Decomate II session identifier can be added if concerns are voiced, and annotations can be submitted through a standard Decomate II Broker session.

To prevent extensive code lists to be hard-coded in the associated programs for internal readability reasons and user feedback (e.g., for translating link codes to semantic link names), and to retain sufficient flexibility in Lexicon structure, a lookup table should be uploaded to any Lexicon client on request. Typically this is done on initial contact. All programs should base their behavior on this lookup table and translate the short transmission codes to the expanded code directly upon reception of the transmission code. Additionally this extra lookup table facilitates multi-lingual operation.

Effort must be taken to prevent any double transmission of information. Given the main purpose of the Lexicon (to assist in network browsing), new concepts can only be added to the client network by two actions: either by direct retrieval of

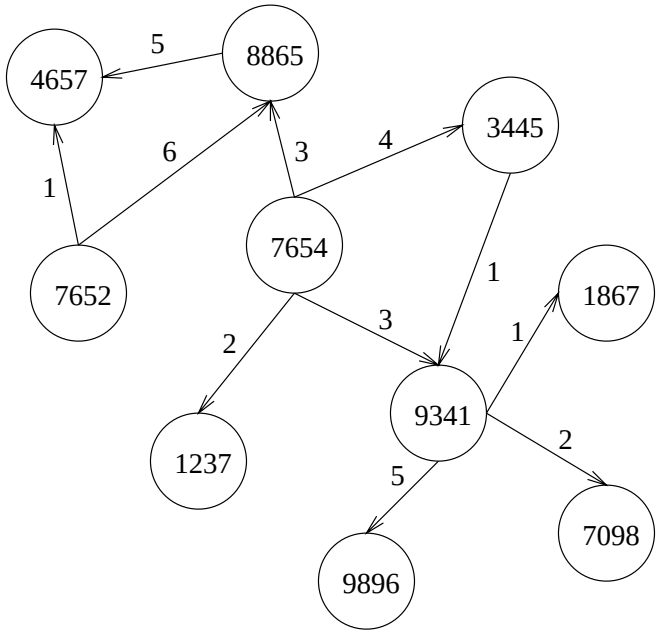


Figure 5.2: A random Lexicon content abstract

concepts through a terminology search, or by expanding a concept to see the concepts surrounding it. The protocol should reflect this situation by allowing all required concept information to be uploaded to the client at initial concept retrieval time. Concept information includes: ID, readable name, and the links leading *away* from the concept.[7] Link information includes link type and the ID of the concept the link leads to. There is a difference between initial (direct) concept retrieval and follow-the-link (indirect) concept retrieval. Directly retrieved concepts should be immediately surrounded by neighboring concepts, i.e., all the neighbors should be retrieved as well, including their readable name and link information. We call this *cascading*. Indirectly retrieved concepts however will not require cascading until the user clicks on them. Because direct retrieval always involves cascading, we include a single level cascade in the protocol to avoid unnecessary network load.

As an example of a typical Lexicon communication session, suppose a user starts with the initial query ‘electronic commerce’ which retrieves two concepts, ‘electronic commerce’ and ‘Internet trade.’ See Figure 5.2 for the associated conceptual network and semantic link codes (meaningless for the moment).

The following example starts after the client has connected to the Lexicon Server; the server send the greeting string and then waits for client requests. The

[7] Although abstract links are bidirectional in nature, actual Lexicon links are directed. A complete link is formed by two paired links of opposite type in opposite directions.

protocol is viewed from the server: > means server transmission, < server reception. Note that there is no explicit 'end of connection' code; the protocol is completely stateless and a client can (and in Web practice, will) quit at any moment.

Important: This is only an example. The actual protocol will be based on Z39.50 and likely follow the XML nested record structure.

```
> Decomate-II Lexicon V1.7
< L
> S 1="broader term",2="narrower term",3="part of", ...
< T "electronic commerce"
> C 7652="Internet trade",1=4657,6=8865; \
    4657="open protocols",2=7652,4=8865,7=5911; \
    8665="Linux",5=4657,2=7654,5=7652,1=5911,7=2243; \
    7654="electronic commerce",3=8865,4=3445,2=1237,3=9341; \
    3445="teleshopping",3=7654,1=7652; \
    1237="amazon.com",1=7654,1=7652; \
    9341="EDIFACT",4=7654,2=3445,1=1867,2=7098,5=9896
< C 8865-4657,7654,7652
> C 5911="open source software",2=8865,1=4455,3=6666; \
    4657="open protocols",2=7652,4=8865,7=5911
```

As can be seen, all relevant concept data is transmitted at 'first contact,' and a term query ('T') immediately invokes a concept cascade. When the user clicks on a specific concept, in the example 8865, the client sends this concept's ID to the Lexicon together with a list of concept IDs of which the client already knows the data. The Lexicon Server then returns the missing data, i.e., links to concepts not already included in the client's network *and linked to the concept clicked on*. As depicted with concept 4657, the returned concept list may include concepts that already are in the client's network but on a different place. It would be impractical to exclude concept data that already is available outside the scope of a single concept cascade, since the exclusion information would easily outgrow the transmitted Lexicon data. The current proposal seems a good compromise.

Also note that the request for lookup table upload ('L') can be done at the client's discretion. If a link gets interrupted, the client can initiate a new link and resume operations without any additional synchronization with the Lexicon Server. A complete client re-initialization for the same reason does not need to re-connect to the Lexicon Server.

Chapter 6

The Concept Mapper

As explained in Chapter 4, the Concept Mapper is an integral part of the Information Space Browser and shares the same interface as the Concept Browser. Workpackage 4 however includes several functions that are technically not suited for including in the Concept Browser. This chapter describes all modules which are part of the post-processing phase of the Decomate II process, whether they are included in the ISB or in the Decomate II back-end.

6.1 Identifying Duplicate Results

Since several bibliographical databases may contain references to the same document, it is likely that one query over multiple databases will turn up duplicated index records. It is unlikely that these records are exactly equal; library experience shows that even in catalogs that are made according to the same indexing rules, significant differences between records may surface. Therefore it is not trivial to detect duplicated records. Nonetheless a good de-duplicator would be very useful to clean out the document presentation to the user.

We cannot even assume that author and title of a document will be represented in duplicated records in exactly the same way, and the other record fields will show a sufficient variance to preclude any comparison. However, records representing the same document will to a certain degree contain the same information—humans would have no significant problem to decide whether or not two records are referring to the same document. We need an algorithm that supports *fuzzy matching* of records, and then be somewhat conservative in rejecting assumed duplicates.

At SilverPlatter Information Ltd, one of the project partners, the same problem has been partially solved by the computation of a hash function that maps congruent document records into the same bucket, thereby indicating a large degree of

overlap. We assume that we can get access to this hash function and will subsequently implement it and assess its applicability to the Decomate II environment.

6.1.1 Place in the System Architecture

It is not trivial whether the de-duplicator should be included in the Broker (or find another place in the back-end), or should be an integrated part of the Advanced Information Browser and thus be included in the Java front-end.

An advantage of including in the back-end is that the de-duplicator then can be used for *all* database queries, and not only for those which are initiated by the Concept Browser. This effectively increases the functionality of the whole system. Disadvantage, obviously, is that this module then must be included after the core functionality has been established, since Workpackage 4 is scheduled for delivery later than 3.

When duplicated records are detected, a decision must be made which record to display and which to hide. It seems logical to give priority to records for which the corresponding document is available in the local library. However, it might happen that the record from a remote library is much more extensive. Electronic documents might be available both locally and remotely—possible download costs then come into play.

The pragmatic solution is to display the most extensive record, or even to aggregate information of several records into one compilation. Any availability of electronic versions of the document should be indicated. When a user wants the electronic document, the available options should be shown with their price and possibly delivery time, or 'network distance.' Physical documents' locations can be shown at the same time. This separation between index record and actual documents should not really hamper the user, and offering some choice in storage locations is a logical concept.

Since this choice must be made by the user, it is inevitable to upload the document availability information to the user's browser at some point. The decision where to put the de-duplication code influences this moment in time. When the de-duplicator is put inside the Concept Browser (in the user's Web browser), all document availability information must be uploaded at query answer time, to build the appropriate user interface. When the de-duplicator is built into the Broker, the availability information can be held back until the user actually requests it. Since it is nowhere certain that every document index record that is turned up by the database back-end will actually be of interest, and since retrieving all this information can be time-consuming, it seems best to delay the uploading of document location information for as long as possible.

Other reasons exist that favor a place in the back-end, e.g., the tight coupling between Document Servers and the back-end (for security and accounting

control), and policy decisions about document source availability, possibly even including network outages. It is much more convenient to have this information centralized in the back-end than included in code for the front-end.

Summarizing, we decide that the best place for the de-duplicator is in the back-end. This sets the de-duplicator apart from the Advanced Information Browser. It should be considered a separate module, that 'by coincidence' ended up in Work-package 4. Implementation should be done in the same way as for the rest of the back-end, i.e., in C++, and the module should be part of the regular version control system of the back-end.

6.2 Document Location Discovery

Even more fuzzy than the de-duplication described in Section 6.1 is the task of locating *electronic* versions of a document. It may be the case that after de-duplication, the most extensive document index record has been presented to the user while another DIR contained a reference to an electronic document. Or that a record has been retrieved from a database that did not contain electronic references,[1] while there is an electronic copy available from an unrelated database. Both situations clearly are undesirable.

Another scenario could be that several electronic documents are available, but not all at the same price. One document might be part of a university collection, which means that it is for free for people affiliated to that university but unavailable to others. The same document might also be available on the basis of pay-per-view directly from the publisher, which makes it unattractive for university staff members with a local subscription but the only alternative for company users.

The most sensible thing to do seems to search in one or more article databases with location information that are in the Decomate domain. The problem is how to create a query that has a very high probability of finding the indexing record in such an article database. For specific publishers, using their own document code might be a good way out. Of course, the real solution will be that all databases use the same universal document identifiers, but that is not to be expected in the near future.

In any case, a solution to this problem is much more likely to come from electronic commerce research than from information retrieval research. Many projects concentrate on information brokering, and document brokering is based on the same principles. Eventually, external intermediate brokers might be in-

[1] This can happen when a user specifically restricts the search to a few databases.

volved to actually solve the problem for the individual user (SilverPlatter Ltd. comes to mind here). In any case, electronic document source suggestions must be finely tuned to the user and his/her environment. Generic solutions probably are not feasible; it seems more likely that every library system needs its own set of options and rules.

As with the de-duplication, the correct place for the document discovery module probably is the back-end and not the Concept Browser. Much of the required information will be available from the individual databases and from the Decomate II Meta-Database, and only the back-end has the specific user information such as access rights readily available.

6.3 Result Ranking

An essential task of the Information Space Browser is to present the retrieved document index records in a certain sort order. The database back-ends often offer some explicit sorting, but these sort orders are only relative to their own output. Some post-processing (merging) by the Concept Mapper is inevitable. Merging is only possible on a relative scale, such as the order of the alphabet, with deterministic 'positions' of the various document records.

The ranking most wanted is by relevance, with the most relevant document index record on top of the list. For obvious reasons this is by no means trivial. Many databases in Decomate II offer some form of relevance ranking, but the used algorithms are not always known (Zebra) and the various algorithms are most likely not mapped onto the same scale. It would be impossible to calculate the relevance ratio of documents coming from separate databases—the number 1 relevant document of one database might very well be significantly less relevant than the number 10 relevant document of another database.

For practical purposes we therefore must assume that all results returned by the back-end are *unsorted*, and do all the sorting ourselves. A workaround could be to present the results *per database* and keep the database sort order.

6.3.1 Relevance Ranking

There are some obvious ways to rank documents that might be of interest to users. One is on reversed record entry time, with the newest document on top. Alphabetical author ordering, or grouping by publisher, might be useful to some, as is grouping by availability (electronically, locally, external order). The main ordering of interest however is *relevance ranking*, with the most relevant document on top.

Of course relevance ranking is the main topic of much cutting-edge research, and despite many years of research no universal solution has been found yet. In Decomate II, we cannot rely on heavyweight statistics or other computationally intensive methods requiring an extensive full-document database, which rules out most of the current scientific approaches.

The only data which we can use for relevance ranking in Decomate II is the actual content of the retrieved document index records—the real documents are only rarely available in electronic format, and often as scan (TIFF) files, unsuitable for processing. If this contents does not contain an abstract, as will often be the case, it is an almost hopeless task to provide any ranking at all. Other ranking systems, such as on author name, would still work well under such circumstances.

Given a list of query results with abstracts, to provide any ordering we still need some reference of what would be the most relevant document. The plain Boolean keyword query as sent to the back-end would be too sparse a source for such a reference. But given that the Boolean query has been compiled by the front-end (Concept Browser), much more information about the user's wishes is known. This leads to communication between the two halves of the front-end, the Concept Browser and Concept Mapper, see Figure 3.2 on page 8. See Section 6.5 for a discussion of Relevance Feedback.

As described in Section 4.3.5, it seems a good idea to move at least *some* database record filtering to the result ranking module. Queries containing 'and' and 'not' operators might be too strict in throwing away database hits. Moving these operators from the hard, Boolean database query to the softer relevance ranking module will cause a lower ranking of document records that contain the limiting words, yet they are still in the result list. If other words in the document records favor a high relevance ranking, they will still end up high on the list.

6.4 Mapping Query Results into the Browser

To give a good indication to the user which concepts in his or her query seem to be related to retrieved documents, it is best to show the index records as clickable dots on the concept map itself.

A trivial way to implement this is to show the document with all the concepts it shares a keyword with. This leads to multiple appearance of most documents, and gives no further clues about which concept(s) it matches best. Mathematically it is possible to calculate the single point in concept space where the document would be at the conceptual center of gravity (cf. the vector model in IR research), but that would mean almost nothing to the user because the concept space itself might (will) be mostly unordered. In practice, we suppose a combined approach will work best: showing the record in a few concepts where it seems to fit best.

Optionally the user could be offered the feature to 'collapse' all concepts related to a record into one complex concept, which is from then on used as a single unit with which queries can be conducted. The explicit availability of an index record, possibly including an abstract, might further enhance the semantic content of the new concept and increase the concept's value in relevance feedback.

6.5 Relevance Feedback

Given that a user has compiled a query and that the back-end has returned the search results, the Concept Mapper can update the network map with the retrieved document index records. In order not to clutter up the map with meaningless speckles representing documents, we indicate successful[2] concept matches by a simple graphical change of the concept. In Figure 6.1 the circle surrounding the concept dot signals the availability of DIRs. The size of the circle could be an indication of either the number of matches, or the calculated 'quality' of the match.

By left-clicking on the encircled concept the user asks for a breakdown of the retrieved documents associated with the concept. In case the concept has not yet been cascaded, this is being done at the same time. The document index record list is shown to the user in the form of a pop-up window, not disturbing the layout of the underlying network map.

The DIR list is closely comparable to a standard Decomate II result list, ordered on relevancy, with one difference. The user can check the individual DIRs as being 'relevant' or 'not relevant' and re-submit the network again. Because essentially the user now has given more terminology information to the network, the Concept Browser should be able to compile a better Boolean keyword query, using terms available in the DIRs of both the relevant and the irrelevant documents.

Alternatively the newly found terminology could be used to expand or refine the current network, leading to the addition or deletion of concepts, which in turn would bring in more terminology (negative and positive). A combination of direct query modification through the document's terminology and through concept terminology is also possible.

It is especially this aspect of the Advanced Information Browser that should be carefully researched; relevance feedback is recognized as one of the most effective forms of query refinement, but over-enthusiastic use of the mechanism can easily lead to a decrease in information retrieval effectiveness.

[2]Of course, it is a matter of statistics to decide whether a retrieved document truly matches a concept or not, unless documents are explicitly connected to concepts, e.g., by documentalists.

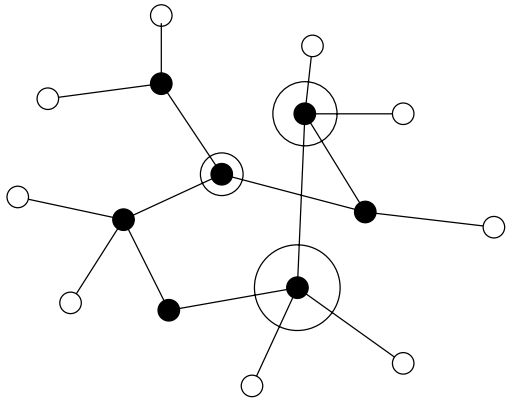
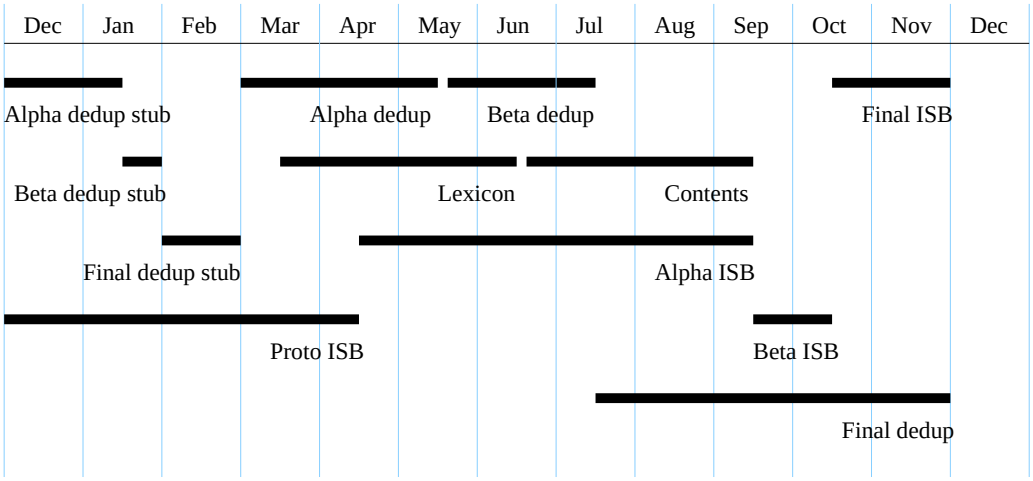


Figure 6.1: Mapping of records into the network

Chapter 7

Implementation Schedule

Dec	1998	Start of Implementation
Jan	1999	Alpha release of de-duplication and doc tracing stubs
Feb	1999	Beta release of de-duplication and doc tracing stubs
Mar	1999	Final release of de-duplication and doc tracing stubs
Apr	1999	Prototype of Information Space Browser in Tcl/Tk
May	1999	Alpha release of de-duplication and doc tracing
Jun	1999	Lexicon implementation in Zebra, existing thesauri only
Jul	1999	Beta release of de-duplication and doc tracing
Sep	1999	Specialized Lexicon content
Sep	1999	Alpha release of Information Space Browser in Java
Oct	1999	Beta release of Information Space Browser
Nov	1999	End of Implementation



Bibliography

Aitchison, J. and Gilchrist, A. (1987). *Thesaurus Construction*. Aslib, London. 2nd edition.

Aslandogan, Y., Thier, C., Yu, C. T., Zou, J., and Rishe, N. (1997). Using Semantic Contents and WordNet in Image Retrieval. In Belkin, N. J., Narasimhalu, A. S., and Willett, P., editors, *Proceedings of the 20th ACM SIGIR Conference*, pages 286–295.

Blair, D. and Maron, M. (1985). An evaluation of retrieval effectiveness for a full-text document retrieval system. *Communications of the ACM*, 28(3):4–22.

Bodner, R. and Song, F. (1996). Knowledge-based approaches to query expansion in information retrieval. In *Lecture Notes in Computer Science*, volume 1081, pages 146–158.

Carbonell, J. and Thomason, R. (1986). Parsing in biomedical indexing and retrieval. In *AAMSI-86*.

Chen, H. (1995). Machine Learning for Information Retrieval: Neural networks, Symbolic Learning and Genetic Algorithms. *Journal of the American Society for Information Science*, pages 194–216.

Cooper, J. W. and Byrd, R. J. (1997). Lexical Navigation: Visually Prompted Query Expansion and Refinement. In Allen, R. B. and Rasmussen, E., editors, *Proceedings of the 2nd ACM International Conference on Digital Libraries*. `http://www.ibm.research.com/people/j/jwcnmr/`.

de Cock, R. (1998). Frankenstein Returns. Personal communication, Decomate-II Project.

Fitzpatrick, L. and Dent, M. (1997). Automatic Feedback using Past Queries: Social Searching? In Belkin, N. J., Narasimhalu, A. D., and Willett, P., editors, *Proceedings of the 20th ACM SIGIR Conference*, pages 306–313.

Gruber, T. (1993). A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220.

Hoppenbrouwers, J. (1997). *Conceptual Modeling and the Lexicon*. PhD thesis, Tilburg University. `http://infolab.kub.nl/people/hoppie`.

Hoppenbrouwers, J. (1998). Browsing Information Spaces. In Prinsen, J., editor, *International Summer School on the Digital Library 1998*, Tilburg, The Netherlands. Ticer B.V.
`http://infolab.kub.nl/people/hoppie`.

Howard, H. (1992). Measures that discriminate among online searchers with different training and experience. *Online Review*, 6:315–327.

Jansen, B. J., Spink, A., Bateman, J., and Saracevic, T. (1998). Real life information retrieval: A study of user queries on the web. *SIGIR Forum*, 32(1):5–17.

Kaminsky, J. (1969). *Language and Ontology*. Southern Illinois University Press.

Lancaster, F. (1986). *Vocabulary Control for Information Retrieval*. Information Resources Press, Arlington VA.

Lesk, M. (1998). "Real World" Searching Panel at SIGIR '97. *SIGIR Forum*, 32(1):1–4.

Maulding, M. L. (1991). *Conceptual Information Retrieval: a case study in adaptive partial parsing*, volume 152 of *Kluwer international series in engineering and computer science*. Kluwer Academic Publishers.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. (1993). Introduction to wordnet: An on-line lexical database. Technical report, Princeton University.

Miller, U. (1997). Thesaurus Construction: Problems and their Roots. *Information Processing and Management*, 33(4):481–493.

Milstead, J. (1992). Methodologies for subject analysis in bibliographic databases. *Information Processing and Management*, 28:407–431.

Ousterhout, J. K. (1994). *Tcl and the Tk Toolkit*. Addison-Wesley.

Papazoglou, M. (1997). Knowledge Navigation and Information Agents: Problems and Issues.

Papazoglou, M., Weigand, H., and Milliner, S. (1998). TopiCA: A Semantic Framework for Landscaping the Information Space in Federated Digital Libraries.

Parsaye, K., Chignell, M., Khoshafian, S., and Wong, H. (1989). *Intelligent Databases: Object-Oriented, Deductive Hypermedia Technologies*. John Wiley and Sons, New York, NY.

Robinson, M. and Bannon, L. (1991). Questioning representations. In Bannon, L., Robinson, M., and Schmidt, K., editors, *Proceedings of the Second European Conference on Computer-Supported Cooperative Work*, page 219.

Salton, G. (1991). Developments in Automatic Text Retrieval. *Science*, 253.

Salton, G. and Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297.

Salton, G. and McGill, M. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY.

Schatz, R. et al. (1996). Interactive Term Suggestion for Users of Digital Libraries. In *1st ACM International Conference on Digital Libraries*, pages 126–133. Bethesda. MD.

Shaw, W., Burgin, R., and Howell, P. (1997). Performance standards and evaluations in IR test collections: vector space and other retrieval models. *Information Processing and Management*, 33(1):15–36.

Sowa, J. F. (1983). *Conceptual Structures, information processing in mind and machine*. Addison-Wesley, Reading, Massachusetts.

Ulijn, J. M. and Strother, J. B. (1995). *Communicating in Business and Technology: From Psycholinguistic Theory To International Practice*. Peter Lang GmbH, Frankfurt.

Vossen, P. (1997). EuroWordNet: a multilingual database for information retrieval. In *Proceedings of the DELOS workshop on Cross-language Information Retrieval, March 5-7, 1997, Zürich*.

Vossen, P., Diez-Orzas, P., and Peters, W. (1997). The Multilingual Design of the EuroWordNet Database. In *Proceedings of the IJCAI-97 workshop Multilingual Ontologies for NLP Applications, August 23, 1997, Nagoya*.

Weigand, H. (1990). *Linguistically Motivated Principles of Knowledge Base Systems*. Foris Publications, Dordrecht, Holland.

Wiederhold, G. (1995). Value-added mediation in large-scale information systems. In *Database Applications' Semantics, IFIP TC-2 DS-6.*

Wiesman, F. (1998). *Information Retrieval by Graphically Browsing Meta-Information*. PhD thesis, Maastricht University, The Netherlands.

Woods, W. A. (1997). Conceptual Indexing: A Better Way to Organize Knowledge. Technical report, Sun Microsystems Laboratories.
`http://www.sunlabs.com/technical-reports/1997/abstract-61.html`.